Jurnal Rekayasa Elektrika

VOLUME 11 NOMOR 4 AGUSTUS 2015

JRE	Vol. 11	No. 4	Hal 123–156	Banda Aceh, Agustus 2015	ISSN. 1412-4785 e-ISSN. 2252-620x

Analisis Kinerja MySQL Cluster Menggunakan Metode Load Balancing

Taufiq Abdul Gani, Aulia Arafat, dan Melinda
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala
Jl. Tgk. Syech Abdurrauf No. 7, Banda Aceh 23111
e-mail: topgan@unsyiah.ac.id

***Abstrak*—Sistem informasi dan data sudah menjadi hal yang sangat penting saat ini. Suatu sistem informasi khususnya *database* dirancang dengan ketersediaan data yang tinggi. Salah satu sistem *database* yang memiliki fungsi tersebut adalah *MySQL cluster*. Namun ada saatnya performansi dari sistem tersebut mengalami penurunan yang disebabkan oleh ketidakmampuan sistem melayani data yang banyak. Untuk mengatasi hal tersebut dibutuhkan suatu penelitian untuk menambahkan sistem *load balancing* pada *MySQL cluster*. Penelitian ini bertujuan untuk menganalisis kinerja *MySQL cluster* dalam kondisi *default* dan *load balancing*. Analisis dilakukan melalui pengambilan data dan uji coba transaksi *database* yang menitikberatkan pada jumlah *Transaction per second* (TPS) dan *response time* menggunakan aplikasi *SysBench*. Kemudian uji coba tersebut terbagi menjadi 2 skenario utama yaitu *simple-mode* dan *complex-mode* yang dijalankan pada 4 unit komputer. Pengujian dilakukan dengan memberikan beban 8 hingga 128 *threads* kepada *server* melalui *client* pada kondisi *default* dan *load balancing*. Adapun hasilnya adalah TPS pada *MySQL cluster load balancing* dengan nilai 2261,22 lebih baik dibandingkan *MySQL cluster default* dengan nilai 536,61. Sedangkan *response time* pada *MySQL cluster load balancing* dengan nilai 147,55 ms lebih cepat dibandingkan *MySQL cluster default* yang bernilai 335,00 ms.**

Kata kunci: ***load balancing, MySQL cluster, response time, threads, transaction per second***

***Abstract*—Information system and data is a very important thing nowadays. An information system especially a database is designed with a high data availability. One of the database systems is MySQL cluster. Sometimes, its performance decreased because of the system's incapablity to handle the data. To overcome this problem, the research of load balancing systems for MySQL cluster is needed. This research is intended to analyze the effect of load balancing clustering methods on MySQL cluster which by default has been set as failover clustering. The analysis is done through data collection and database transactions test to get the number of Transactions per second (TPS) and response time parameter using SysBench. The test is divided into two main scenarios, the simple and complex-mode that running on 4 computers. The tests were conducted by giving workloads from 8 up to 128 threads respectively to the server through a client on default conditions and load balancing. The test results showed the number of TPS at MySQL cluster default is 536.61. In contrast, number of TPS at load balancing is better at 2261.22. Likewise, the response time at MySQL cluster load balancing is 147.55 ms faster than that of the default one at 335.00 ms.**

Keywords: ***load balancing, MySQL cluster, response time, threads, transactions per second***

I. PENDAHULUAN

Kebutuhan terhadap sistem informasi dan data saat ini sudah menjadi hal yang sangat penting. Suatu sistem informasi khususnya *database* harus selalu bekerja menyediakan data secara terus-menerus sehingga setiap pengguna dapat memperoleh data tersebut tanpa mengalami masalah. Namun, ada saatnya sistem *database* tersebut mengalami gangguan atau kegagalan sehingga dalam beberapa saat pengguna tidak dapat mengakses sistem tersebut.

Untuk mengatasi hal tersebut diperlukan suatu metode *clustering* yang dapat melakukan replikasi data pada beberapa *server*. Salah satu sistem yang menyediakan fasilitas tersebut adalah *MySQL cluster* yang terdiri dari gabungan beberapa *MySQL Server*. Gabungan tersebut membentuk suatu fungsi yang dapat melakukan *failover* dan menjamin 99,99% ketersediaan *database* sehingga ketika salah satu *server* mengalami masalah, fungsi tersebut akan tetap dapat diakses oleh *client* [1].

Walaupun demikian, performansi *MySQL cluster* akan mengalami penurunan dibandingkan dengan *MySQL Server* [2]. Hal ini terjadi karena arsitektur pada *MySQL cluster* secara *default* digunakan sebagai *failover* yang membutuhkan waktu dalam proses pengecekan dan pemindahan koneksi menuju *server* lainnya ketika salah satu *server* mati atau mengalami kerusakan.

Untuk itu diperlukan penambahan *load balancing* yang memungkinkan *database* bekerja secara lebih stabil dan seimbang dengan membagi koneksi dan beban kerja pada beberapa *server*. Penelitian ini menitikberatkan pada parameter penting untuk mengukur kinerja dari *database server* antara lain *Transaction per Second* (TPS) dan *response time* yang diuji pada *simple* dan *complex-mode.*

ISSN. 1412-4785; e-ISSN. 2252-620X
DOI: 10.17529/jre.v11i4.2358

II. STUDI LITERATUR

A. *Definisi Clustering*

1. *Failover Clustering*

Secara ilmu komputer, *failover* adalah proses perpindahan koneksi secara otomatis suatu *server* ketika *server* yang lainnya mengalami masalah atau kerusakan atau dapat juga dikatakan sebagai duplikasi dan *backup data* [3].

Proses ini dapat terjadi tanpa menghentikan proses yang sedang berjalan sehingga perpindahan yang terjadi dilakukan oleh sistem *cluster* itu sendiri. Jenis *clustering* ini lebih ditujukan untuk duplikasi data agar *client* tetap dapat mengakses layanan informasi data walaupun sebenarnya pada sisi *server* sedang terjadi gangguan.

2. *Load Balancing Clustering*

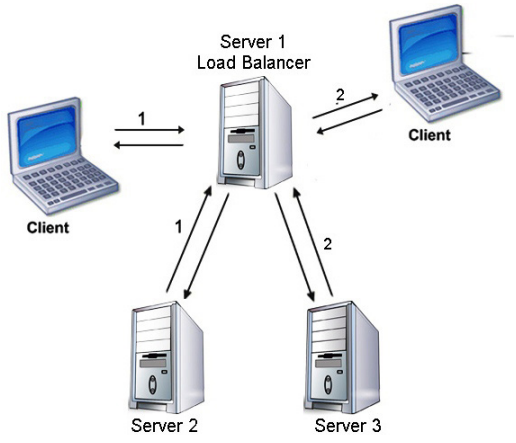
Load balancing merupakan suatu teknologi pembagian trafik internet atau proses kerja pada beberapa *server* dengan menggunakan *hardware* maupun *software* dalam suatu jaringan komputer dan dapat dijalankan sebagai *router* [4].

Sistem *clustering* ini berfungsi untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang sehingga trafik dapat berjalan secara optimal dengan *response time* yang kecil dan proses yang optimal. Ilustrasi sederhana mengenai sistem ini dapat dilihat pada Gambar 1.

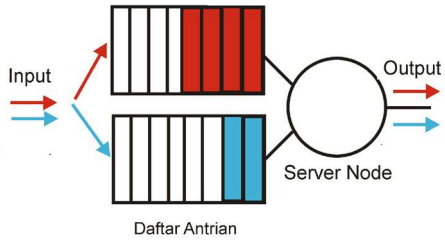
Dari Gambar 1, *load balancer* akan menerima sebuah trafik dari *client* dan kemudian trafik tersebut dilanjutkan ke beberapa *server* lainnya. Trafik tersebut akan diurutkan berdasarkan permintaan *client* dan kondisi *server*. Ketika hanya satu *client* saja yang melakukan request, maka proses yang terjadi hanya pada salah satu *server* saja. Sedangkan apabila melebihi, maka *load balancer* akan membagi proses tersebut secara seimbang untuk masing-masing *server*.

B. *Algoritma Round Robin*

Round Robin adalah salah satu algoritma yang digunakan pada metode *load balancing* untuk membagi beban kerja. Dengan algoritma ini proses dibagi secara merata pada semua *server* yang saling terhubung. Setiap proses baru yang ditugaskan pada *server* akan masuk antrian dan urutannya disusun berdasarkan proses yang sedang berlangsung [5].



Gambar 1. Mekanisme *load balancing clustering* [4]



Gambar 2. Mekanisme algoritma *round-Robin* [5]

Keuntungan dari algoritma ini adalah proses pada *server* menjadi lebih seimbang dan memiliki keseragaman waktu. Proses yang terjadi pada algoritma ini daat dilihat pada Gambar 2.

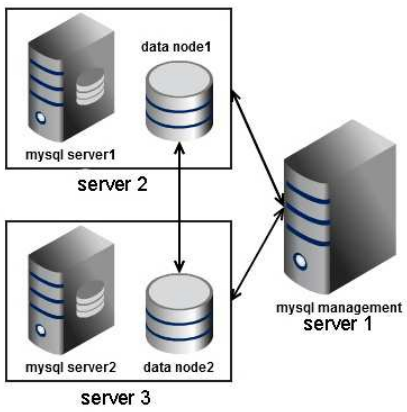
C. *MySQL Cluster*

Tiga *node* utama yang merupakan komponen *MySQL cluster* antara lain [1]:

1. Sepasang Data *Node* yang digunakan untuk menyimpan semua data transaksi pada *MySQL cluster* dan data tersebut direplikasi pada *node* ini.
2. *Management Server Node* digunakan untuk mengendalikan proses pada sistem ketika dihidupkan. Selain itu, *node* ini juga digunakan untuk mengidentifikasi setiap perubahan konfigurasi.
3. *MySQL Server API Node* merupakan *interface* yang digunakan oleh *client* untuk mengakses data dan memanipulasi data pada sistem *cluster*.

D. *SysBench*

SysBench adalah aplikasi *benchmark* yang ditujukan untuk mengukur performansi dari suatu *database* yang dijalankan melalui *command prompt* atau terminal pada sistem operasi linux. Salah satu pengukuran *database* tersebut adalah pengukuran *Online Transaction Processing*



Gambar 3. Komponen utama *MySQL cluster* [1]

(OLTP) yaitu suatu sistem yang melakukan suatu transaksi melalui komputer yang terhubung dalam jaringannya [6].

E. Parameter Pengukuran

1. Transaction per Second

Transaction per Second (TPS) merupakan kemampuan *database* dalam melayani transaksi yang dibutuhkan *client* dalam hitungan detik. Transaksi adalah suatu proses *read* dan *write* yang dilakukan oleh *client* dalam suatu *database*. Proses ini meliputi perintah dan bahasa SQL yang dapat menyebabkan nilai dari tabel atau *database* mengalami perubahan [7].

2. Response time

Response time juga dijadikan sebagai salah satu parameter pengukuran OLTP pada *SysBench*. *Response time* dapat didefinisikan sebagai waktu yang dibutuhkan oleh sistem ketika *client* atau pengguna layanan mengirim permintaaan menuju *server*. Parameter ini tidak dapat ditentukan secara pasti kecuali melalui pengukuran karena dipengaruhi oleh beberapa faktor antara lain kecepatan prosesor, jumlah memori serta interkoneksi sistem [8].

3. Threads

Threads adalah suatu proses *computing* yang terdiri dari suatu data dan instruksi yang dijalankan secara paralel dan terpisah yang akan mempengaruhi kemampuan *server* dalam mengolah data. *Server* akan menerima beban kerja yang dihasilkan oleh instruksi-instruksi tersebut dan akan berubah tergantung modus uji yang telah digunakan [9].

III. Metode Penelitian

Metode yang digunakan pada penelitian ini dapat dilihat pada diagram alir pada Gambar 4. Proses penelitian diawali oleh studi literatur yang dilakukan dengan mengumpulkan berbagai teori-teori pendukung beserta konsep yang berhubungan dengan penelitian ini.

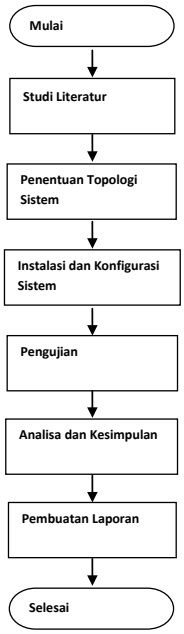
Selanjutnya ditentukan topologi yang sesuai dengan kondisi dan tujuan penelitian. Topologi yang digunakan adalah topologi minimum dari *MySQL cluster* seperti diperlihatkan pada Gambar 5.

Kedua sistem *default* dan *load balancing* menggunakan topologi yang sama. Namun yang membedakannya adalah penambahan sistem *load balancer* pada management *node* untuk membagi beban yang diberikan oleh *client*.

Dari topologi tersebut kemudian dilakukan instalasi sistem *MySQL cluster* pada 3 unit *server* yang masing-masing berfungsi sebagai management *node*, data *node*, SQL-API *node* dan ditambah 1 *client*. Selanjutnya dilakukan instalasi aplikasi *SysBench* khusus pada *client* yang digunakan sebagai media pengujian. Spesifikasi lengkap sistem yang digunakan dapat dilihat pada Tabel 1.

A. Pengujian

Proses pengujian dilakukan pada *client* dengan



Gambar 4. Diagram alir metode penelitian

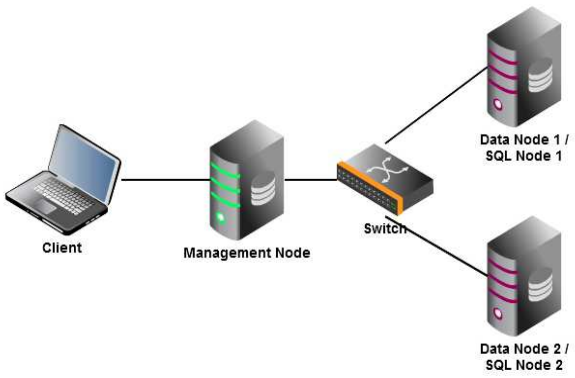
menjalankan aplikasi *SysBench*. Parameter yang diukur adalah TPS dan *response time* yang diubah jumlah *threads*-nya. Perubahan *threads* tersebut dilakukan mulai dari 8, 16, 32, 64, 80, 96, 112 dan 128. Pengukuran tersebut dilakukan berdasarkan mode pengujian berikut:

- *Simple-mode* merupakan pengujian yang dilakukan hanya pada proses *read-only data* atau hanya pembacaan data saja.
- *Complex-mode* merupakan pengujian yang dilakukan dengan proses yang kompleks yang meliputi READ, INSERT, DELETE, dan UPDATE

B. Analisis

Data yang diperoleh kemudian dikumpulkan dan disusun menjadi 2 mode utama yaitu *simple-mode* dan *complex-mode* dan diambil nilai rata-ratanya. Kemudian data-data tersebut dibuat menjadi grafik dan selanjutnya dianalisa menggunakan *t-test* melalui langkah-langkah:

- Ditentukan terlebih dahulu nilai mean dari parameter kondisi *default* dan *load balancing*.



Gambar 5. Topologi pengukuran MySQL *cluster default* dan *load balancing* [4]

Tabel 1. Spesifikasi pengujian *MySQL cluster*

Sistem	Tipe	Prosesor	Memori (MB)	Sistem Operasi	Konektivitas
Server	Lenovo 7515RP4	Pentium® Dual-Core E5400 @2,70 Ghz	1024	CentOS 6.3 i386	D-Link 1008D Switch
Client	Acer 4920	Pentium® Dual-Core T5550 @1,83 Ghz	1536	CentOS 6.3 i386	D-Link 1008D Switch

- Ditentukan pula nilai standar deviasi masing-masing sampel .
- Ditentukan hipotesis acuan sehingga nilai *t-test* dari kedua kondisi dapat dinyatakan sama atau berbeda apabila:
  H_0 : $D = 0{,}05$ kedua kondisi dinyatakan sama.
  H_1 : $D < 0{,}05$ atau $D > 0{,}05$ kedua kondisi dinyatakan berbeda.
- Dihitung nilai *t-test* dengan menggunakan persamaan:

$$t = \frac{\overline{x_1 - x_2}}{S}\sqrt{N} \quad (1)$$

Keterangan:
t : uji *t-test*
x_1-x_2 : nilai rata-rata selisih sampel pertama dan kedua
S : standar deviasi
N : jumlah sampel

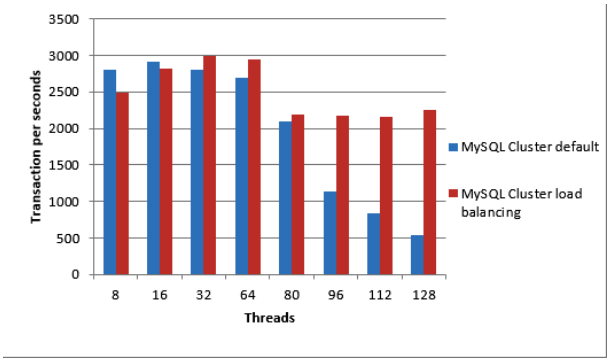
IV. Hasil dan Pembahasan

A. Skenario Simple-mode

Pada mode ini akan ditampilkan data TPS dan *response time* dari hasil transaksi *read-only* yang selengkapnya dapat dilihat pada Tabel 2. Dari tabel tersebut diperoleh penurunan jumlah TPS mulai dari 8 hingga 128 *threads*. Kondisi ini berpengaruh terhadap parameter *response time*s yang semakin lambat akibat penambahan *threads* tersebut. Dari nilai-nilai yang diperoleh pada tabel dapat disusun suatu perbandingan grafik TPS yang dapat dilihat lebih jelas pada Gambar 6.

Tabel 2. Rata-rata pengujian *MySQL cluster simple-mode*

Threads	Default		Load Balancing	
	TPS	Response Time (ms)	TPS	Response Time (ms)
8	2808,77	5,17	2482,71	7,03
16	2912,13	11,03	2813,98	15,59
32	2806,26	31,97	2992,46	31,17
64	2701,65	68,99	2944,16	61,63
80	2097,99	88,03	2187,64	103,17
96	1138,49	148,07	2176,89	124,10
112	830,29	210,98	2157,88	131,46
128	536,61	335,00	2261,22	147,55

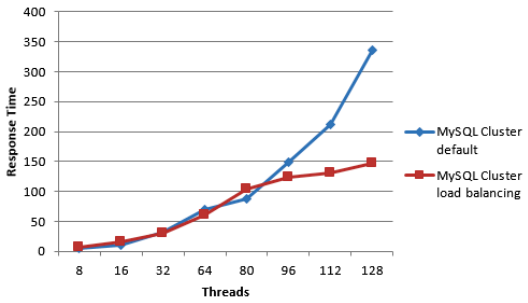


Gambar 6. TPS *simple-mode*

Pada Gambar 6 nilai TPS yang baik akan diperoleh pada nilai yang lebih tinggi. Jumlah TPS yang tinggi menunjukkan kemampuan sistem untuk menyelesaikan transaksi setiap detiknya. Jumlah TPS yang tertinggi adalah 2992,46 pada 32 *threads*. Pada beban 8 dan 16 *threads*, jumlah TPS *MySQL cluster default* lebih tinggi daripada *MySQL cluster load balancing*. Akan tetapi, pada saat 32 hingga 128 *threads* jumlah TPS pada kondisi *load balancing* lebih besar nilainya. Proses ini terjadi karena ketika dilakukan pengujian *database*, transaksi yang terjadi pada kondisi *default* hanya terjadi pada 1 *node* yang aktif (*master*). Sedangkan *node* lainnya berfungsi sebagai duplikasi data yang bekerja apabila master mengalami kegagalan. Berbeda dengan *MySQL cluster load balancing* yang transaksinya terjadi berdasarkan jumlah *node* yang dijadikan *cluster*. Pada kondisi ini terdapat 2 *node cluster SQL* yang akan berbagi beban kerja. Semakin besar thread-nya semakin stabil pula jumlah TPS pada kondisi *load balancing*.

Parameter lain yang berpengaruh terhadap pengujian ini adalah *response time*. Hasil yang diperoleh dari tabel disusun pada Gambar 7.

Gambar 7 memperlihatkan pengaruh penambahan *threads* terhadap *response time*. Parameter ini akan berdampak pada kecepatan akses yang dibutuhkan suatu *node cluster*. Semakin kecil nilai yang diperoleh menyatakan semakin baik pula kondisi *cluster* tersebut. Dapat dilihat bahwa nilai *response time* terbaik adalah 5,17 ms pada saat *MySQL cluster load balancing* diberikan beban 8 *threads*. Sedangkan nilai terburuknya adalah 335 ms pada saat *MySQL cluster default* diberikan beban 128 *threads*. Pada saat diberikan beban 8 dan 16 thread, *response time* pada *MySQL cluster default* lebih



Gambar 7. Response *time simple-mode*

Tabel 3. Rata-rata pengujian *MySQL cluster complex-mode*

Threads	Default		Load Balancing	
	TPS	Response Time (ms)	TPS	Response Time (ms)
8	56,88	181,83	43,29	274,95
16	53,42	424,70	47,72	573,56
32	52,52	823,77	49,17	1070,16
64	51,42	1647,29	47,73	1898,20
80	54,59	1876,88	51,96	2301,08
96	39,55	3080,95	48,73	2888,98
112	30,73	9800,03	50,65	3202,76
128	25,73	11900,58	50,32	3576,01

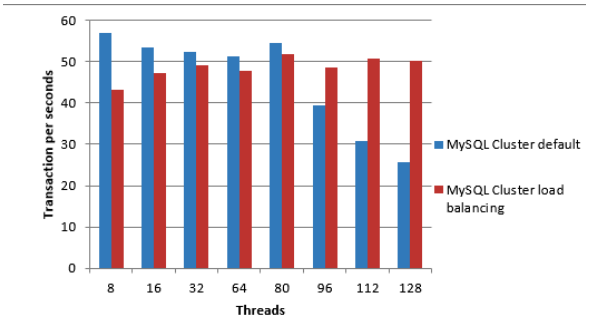
baik dibandingkan *MySQL cluster load balancing*. Namun pada saat beban pada 32 hingga 128 *threads*, *response time* pada *MySQL cluster load balancing* lebih baik. Titik jenuh keduanya *cluster* tersebut berada pada 80 *threads*.

Pada saat itu, nilai perbandingan keduanya mengalami perubahan dan perubahannya semakin signifikan setelah mendekati 128 *threads*. Proses ini dipengaruhi oleh pembagian beban trafik untuk tiap-tiap *node* pada kedua sistem *cluster*. Untuk kondisi *load balancing*, *response time* semakin stabil dan seimbang pada jumlah *threads* besar karena pada saat itu sistem *cluster* membagi beban dari *threads* tersebut ke setiap *node*-nya. Hal ini menyebabkan perbedaan pada *response time* keduanya.

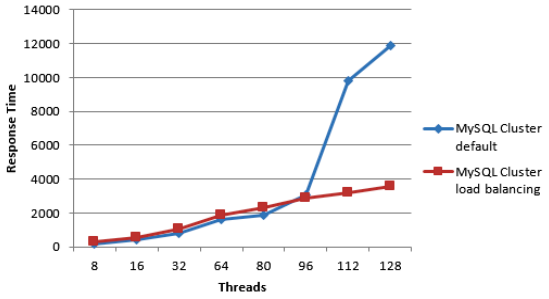
B. *Skenario Complex-mode*

Pada mode ini data yang ditampilkan adalah TPS dan *response time*. Terdapat beberapa perbedaan dibandingkan *simple-mode*. Untuk hasil selengkapnya dapat dilihat pada Tabel 3. Nilai yang diperoleh pada Tabel 3 menunjukkan perbedaan dari Tabel 1. Jumlah TPS pada kondisi *default* mengalami penurunan. Namun, mengalami kenaikan pada kondisi *load balancing*. Nilai tersebut akan naik hingga 80 *threads* dan stabil hingga 128. Ilustrasi perbedaan tersebut dapat dilihat pada Gambar 8.

Pada Gambar 8 dapat dilihat jumlah TPS yang tertinggi adalah 56,88 pada beban 8 *threads*. Sedangkan nilai TPS terendah adalah 25,73 pada beban 128 *threads*. Jumlah TPS yang diperoleh pada mode ini lebih kecil dibandingkan pada *simple-mode*. Hal ini dikarenakan



Gambar 8. TPS *complex-mode*



Gambar 9. Response *time complex-mode*

pada mode ini terjadi variasi proses transaksi SQL yang meliputi; READ, WRITE, INSERT dan UPDATE. Apabila salah satu dari transaksi tersebut tidak dipenuhi, maka transaksi yang terjadi tidak terhitung. Berbeda dengan *simple-mode*, transaksi yang terjadi hanya READ saja. Jadi kemungkinan transaksi yang terjadi semakin banyak. Selanjutnya untuk nilai *response time* dapat dilihat pada Gambar 9.

Perubahan yang terjadi pada Gambar 9 menunjukkan nilai yang terbaik adalah 181,83 ms pada saat kondisi *default* pada beban 8 *threads*. Sedangkan nilai terburuknya adalah 11900,58 ms pada saat kondisi *default* diberikan beban 128 *threads*. Nilai tersebut terlihat sangat signifikan disebabkan pengujian pada *complex-mode* terjadi adanya variasi transaksi yang membutuhkan waktu yang lama untuk menyelesaikannya. Proses tersebut juga dipengaruhi oleh pembagian beban trafik pada sistem *cluster* keduanya. Untuk *MySQL cluster load balancing*, *response time* semakin seimbang pada jumlah *threads* besar karena sistem *cluster* membagi beban dari *threads* tersebut ke setiap *node*-nya.

Kemudian dari hasil rata-rata keseluruhan disusunlah suatu perhitungan untuk mengetahui signifikansi dari penelitian yang telah dilakukan dan diukur menggunakan *t-test* antara *MySQL cluster default* dan *MySQL cluster load balancing*. Hasil dari *t-test* tersebut dapat dilihat pada Tabel 4.

Dari Tabel 4 dapat dilihat bahwa nilai Standar Deviasi (SD) yang diperoleh pada kedua kondisi mengalami perbedaan. Hal ini dikarenakan nilai sampel data yang diperoleh dari pengujian berbeda-beda. Pada kondisi

Tabel 4. Hasil analisa *t-test MySQL cluster*

Threads	Standar Deviasi (SD)		t-test
	Default	Load Balancing	
8	63,12	27,68	4,23774e-21
16	34,56	101,31	3,14310e-05
32	32,91	101,44	2,54458e-11
64	31,18	70,64	2,13516e-19
80	54,26	76,52	1,46231e-06
96	41,26	95,52	1,37283e-31
112	20,31	54,68	3,73206e-37
128	25,66	106,98	6,44069e-39

default nilai SD semakin menurun untuk setiap penambahan *threads*. Sebaliknya pada kondisi *load balancing* nilainya meningkat dan memiliki varian yang berubah-ubah.

Kemudian pada nilai *t-test* digunakan untuk melihat signifikansi antara kondisi *default* dan *load balancing*. Syarat terjadinya signifikansi adalah ditolaknya nilai *Ho* (*t-test* < 0,05). Dari Tabel 4 dapat dilihat bahwa seluruh nilai *t-test* yang diperoleh bernilai dibawah 0,05 atau 5e-02. Oleh karena itu karena syarat *Ho* tidak terpenuhi, maka kedua kondisi tersebut mengalami signifikansi dan memiliki varians yang berbeda.

V. KESIMPULAN

Berdasarkan hasil penelitian, dapat disimpulkan bahwa penurunan performansi yang terjadi pada *MySQL cluster* dapat sedikit ditingkatkan dengan penambahan *load balancing*. Penambahan tersebut dapat berjalan dengan baik pada saat beban diatas 80 *threads*. Jumlah TPS *MySQL cluster load balancing* yang bernilai 50,65 lebih baik dari pada *MySQL cluster default* yang bernilai 25,73. Selanjutnya untuk parameter *response time*s pada *MySQL cluster load balancing* yang bernilai 3576,01 lebih cepat dibandingkan *MySQL cluster default* yang bernilai 11900,58.

REFERENSI

[1] Davies, Alex and Harrison Fisk. MySQL Clustering, MySQL Press, United Stated of America, 2006.

[2] Verdian, Van. Perbandingan MySQL Server dan MySQL Cluster, Tugas Akhir Jurusan Teknik Elektro Universitas Syiah Kuala, Banda Aceh, 2012.

[3] Schwartz, Baron, Peter Zaitsev, Vadim Tkacenko, Jeremy D. Zawodny, Arjen Lentz, and Derek J. Balling. High Performance MySQL. O'Reilly Media, California, 2008.

[4] Membrey, Peter, Eelco Plugge, and David Hows. Practical Load balancing: Ride the Performance Tiger. Apress, New York, 2012.

[5] Xu, Zhong and Rong Huang. Performance Study of Load balancing Algorithms in Distributed Web Server Systems. CS213 Parallel and Distributed Processing Project Report, 2009.

[6] Kopytov, Alexey. SysBench Manual. [Online] Available :. http://SysBench.sourceforge.net/docs, 2009.

[7] Coronel, Carlos, Steven Morris, and Peter Rob. Database Systems: Design, Implementation, and Management, 10th Edition. Cengage Learning, Stamford, 2012.

[8] Inmon, William H. Building the Data Warehouse, Wiley Publishing, Indianapolis, 2005.

[9] Jadhav, S. S. Advanced Computer Architecture and Computing. Technical Publications Pune, India, 2009.

Penerbit:
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala
Jl. Tgk. Syech Abdurrauf No. 7, Banda Aceh 23111
website: http://jurnal.unsyiah.ac.id/JRE
email: rekayasa.elektrika@unsyiah.net
Telp/Fax: (0651) 7554336